Sistema de Recomendação Para Rede de Varejo Usando Market Basket Analysis

Sobre

Projeto com Feedback nº 10 do curso Business Analytics da Formação Cientista de Dados da Data Science Academy.

A partir da análise do dataset "The Instacart Online Grocery Shopping Dataset 2017" - quem contém mais de 3 milhões de pedidos de mais de 200 mil usuários - foi feito um trabalho de análise de dados com o objetivo de prever quais produtos adquiridos anteriormente estarão no próximo pedido de um usuário.

Acesso para o dataset: instacart

Artigo sobre o case: artigo

Baixando pacotes necessários

```
library(data.table)
library(dplyr)
library(arules)
library(ggplot2)
library(gridExtra)
library(wordcloud)
library(xgboost)
library(naniar)
library(caret)
library(RColorBrewer)
library(kableExtra)
```

Leitura dos datasets e Dicionários de Dados

Tabela de Pedidos (tabela Fato) - 3.4 milhões de linhas

```
pedidos <- fread("datasets/orders.csv")

orders_dic <- data.frame(Variável = c("ID Pedido", "ID Comprador", 'Classificação do pedido',
                                      "Número do Pedido", "Dia do Pedido", "Hora do Pedido",
                                      "Dias Desde o Último Pedido"),
                         Descrição = c("identificação do pedido", "identificação do comprador",
"classificação de pedido: prior = pedido comum/
train = pedidos reservados para dataset de treino/
test = pedidos reservados para dataset de teste",
                         "nº do pedido realizado pelo consumidor",
                         "dia da semana que o pedido foi realizado pelo consumidor",
                         "hora do dia que o pedido foi realizado",
                         "dias desde o último pedido deste consumidor"))

orders_dic  %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T) %>%
    column_spec(2, width = "10cm")
```

Variável	Descrição
ID Pedido	identificação do pedido
ID Comprador	identificação do comprador
Classificação do pedido	classificação de pedido: prior = pedido comum/ train = pedidos reservados para dataset de treino/ test = pedidos reservados para dataset de teste
Número do Pedido	nº do pedido realizado pelo consumidor
Dia do Pedido	dia da semana que o pedido foi realizado pelo consumidor
Hora do Pedido	hora do dia que o pedido foi realizado
Dias Desde o Último Pedido	dias desde o último pedido deste consumidor

Tabela de Produtos (tabela Dimensão) - 50 mil linhas

```
produtos <- fread("datasets/products.csv")

produtos_dic <- data.frame(Variável = c("ID Produto", "Nome do Produto",
                                        "ID do Corredor", "ID do Departamento"),
                           Descrição = c("identificação do produto", "nome do produto",
                                         "identificação do corredor que o produto está localizado.",
                                         "identificação do departamento que o produto pertence."))

produtos_dic %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

Variável	Descrição
ID Produto	identificação do produto
Nome do Produto	nome do produto
ID do Corredor	identificação do corredor que o produto está localizado.
ID do Departamento	identificação do departamento que o produto pertence.

Tabela de Corredores (tabela Dimensão) - 134 linhas

```
corredores <- fread("datasets/aisles.csv")

corredores_dic <- data.frame(Variável = c("ID Corredor", "Nome do Corredor"),
                             Descrição = c("identificação dos corredores do mercado.",
                                           "nomes dos corredores do mercado."))

corredores_dic %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

Variável	Descrição
ID Corredor	identificação dos corredores do mercado.
Nome do Corredor	nomes dos corredores do mercado.

Tabela de Departamentos (tabela Dimensão) - 21 linhas

```
departamentos <- fread("datasets/departments.csv")

departamentos_dic <- data.frame(Variável = c("ID Departamento", "Nome do Departamento"),
                                Descrição = c("identificação dos departamentos do mercado.",
                                              "nome dos departamentos do mercado."))

departamentos_dic %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

Variável	Descrição
ID Departamento	identificação dos departamentos do mercado.
Nome do Departamento	nome dos departamentos do mercado.

Tabelas de Especificidades de Pedidos - >= 30 milhões de linhas

Carregando apenas 'order_products___train.csv' para aplicar a regra de associação com ele por conta de memória

```
pedidos_esp <- fread("datasets/order_products__train.csv")

pedidos_esp_dic <- data.frame(Variável = c("ID Pedido", "ID Produto",
                                           "Ordem de Adição ao Carrinho", "Produto Repetido?"),
                              Descrição = c("identificação do pedido", "identificação do produto",
                                            "ordem que o item foi adicionado ao carrinho em determinada
                                            "informa se o produto está sendo pedido novamente (1) ou nã

pedidos_esp_dic %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

Variável	Descrição
ID Pedido	identificação do pedido
ID Produto	identificação do produto
Ordem de Adição ao Carrinho	ordem que o item foi adicionado ao carrinho em determinada compra
Produto Repetido?	informa se o produto está sendo pedido novamente (1) ou não (0).

Tratamento dos datasets

Fazendo cópia dos datasets para partes 1 e 2 (boa prática caso seja necessário voltar ao dataset original por algum motivo)

```
df_pedidos <- pedidos
df_pedidos2 <- pedidos
```

Valores NA

```
na_count <- as.data.frame(sapply(df_pedidos, function(y) sum(length(which(is.na(y))))))
names(na_count) <- "Qtd. NAs"

na_count %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

	Qtd. NAs
order_id	0
user_id	0
eval_set	0
order_number	0
order_dow	0
order_hour_of_day	0
days_since_prior_order	206209

Observamos que existe um número considerável de NAs, porém,apenas na coluna referente a quantidade de dias desde a última compra de determinado consumidor. Para tratar esses valores NAs, vamos substituir

esses NAs pelo número 0, tendo em vista que se não houve intervalo de dias entre as compras, podemos aplicar o número zero para representar esta informação.

```
df_pedidos$days_since_prior_order[is.na(df_pedidos$days_since_prior_order)] <- 0
df_pedidos2$days_since_prior_order[is.na(df_pedidos2$days_since_prior_order)] <- 0
```

Parte 1

Análise Exploratória com identificação de itens mais pedidos em conjunto (duplas) Foco na exploração do dataset com pacote dlpyr e depois aplicação dos algoritmos de MBA (Market Basket Analysis) ECLAT e APRIORI.

Feature Engineering

Criando nova coluna (dia_da_semana) com nome do dia da semana para posterior análise exploratória e a transformando em fator

```
df_pedidos$dia_da_semana[df_pedidos$order_dow == 0] <- "Domingo"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 1] <- "Segunda"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 2] <- "Terça"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 3] <- "Quarta"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 4] <- "Quinta"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 5] <- "Sexta"
df_pedidos$dia_da_semana[df_pedidos$order_dow == 6] <- "Sábado"

df_pedidos$dia_da_semana <- factor(df_pedidos$dia_da_semana,
        levels = c("Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"))
```

Periodicidade de Compras

Se days_since_prior = 0, então nenhuma, ou seja, o usuário não fez nenhum outro pedido na plataforma Se days_since_prior entre 1 e 15 inclusive, então média, ou seja, o usuário fez de 1 até 15 pedidos na plataforma Se days_since_prior > 15 exclusive, então alta, ou seja, o usuário realizou mais de 15 pedidos na plataforma

```
df_pedidos$periodicidade[df_pedidos$days_since_prior_order == 0] <- "baixa"
df_pedidos$periodicidade[df_pedidos$days_since_prior_order > 15] <- "alta"
df_pedidos$periodicidade[is.na(df_pedidos$periodicidade)] <- "média"
```

Tratamento de dataset pedidos_esp para os algoritmos de MBA

O dataset pedidos_esp será combinado com demais tabelas dimensão e será utilizado, além de em alguns momentos para análise exploratória, prioritariamente para utilização no ECLAT e no APRIORI

```
pedidos_esp_analise <- pedidos_esp %>%
    left_join(produtos, by = "product_id")

df_modelo <- pedidos_esp_analise %>%
    group_by(order_id) %>%
    summarise(produtos = as.vector(list(product_name)))
```

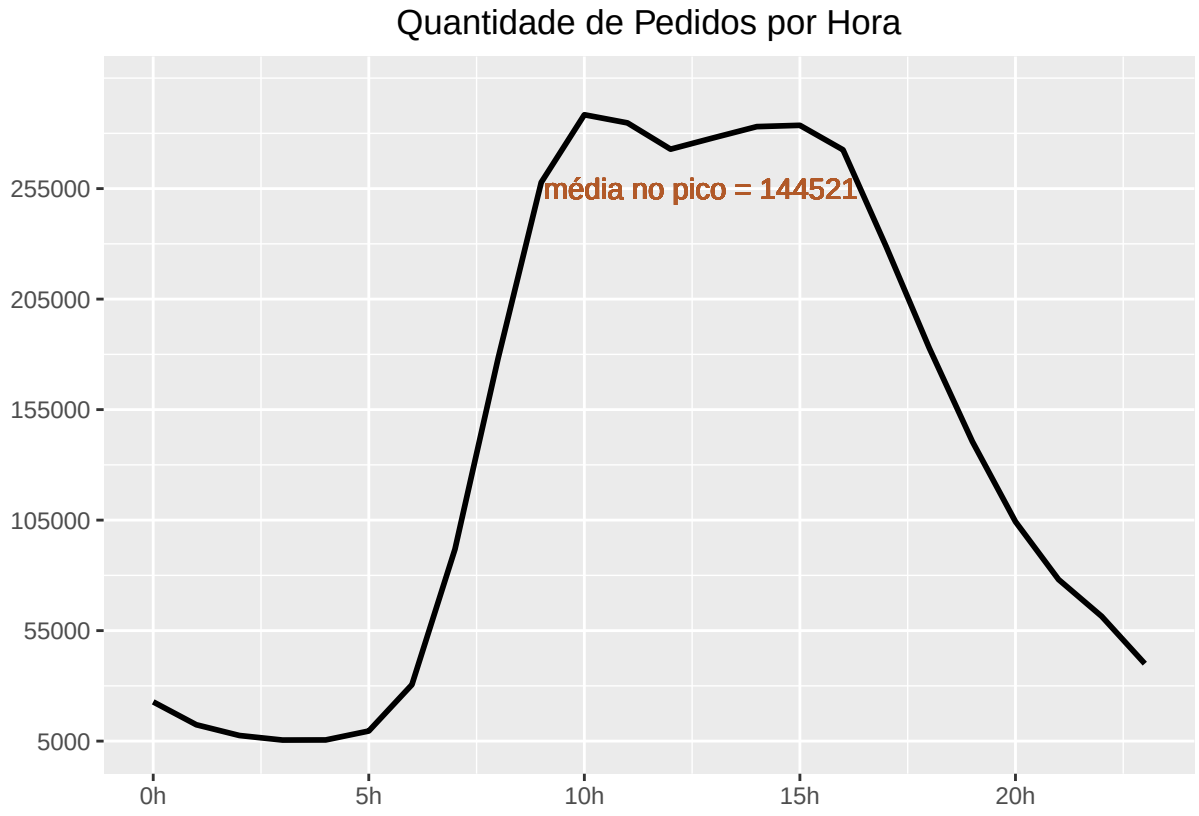
Análise Exploratória

Para construção dos gráficos usarei a escala Greys do pacote RColorBrewer, com o acréscimo de uma cor para ter uma paleta com 10 cores

```
cores <- c(brewer.pal(n = 9, "Greys"),"#999999")
```

Gráfico de quantidade de pedidos por hora do dia

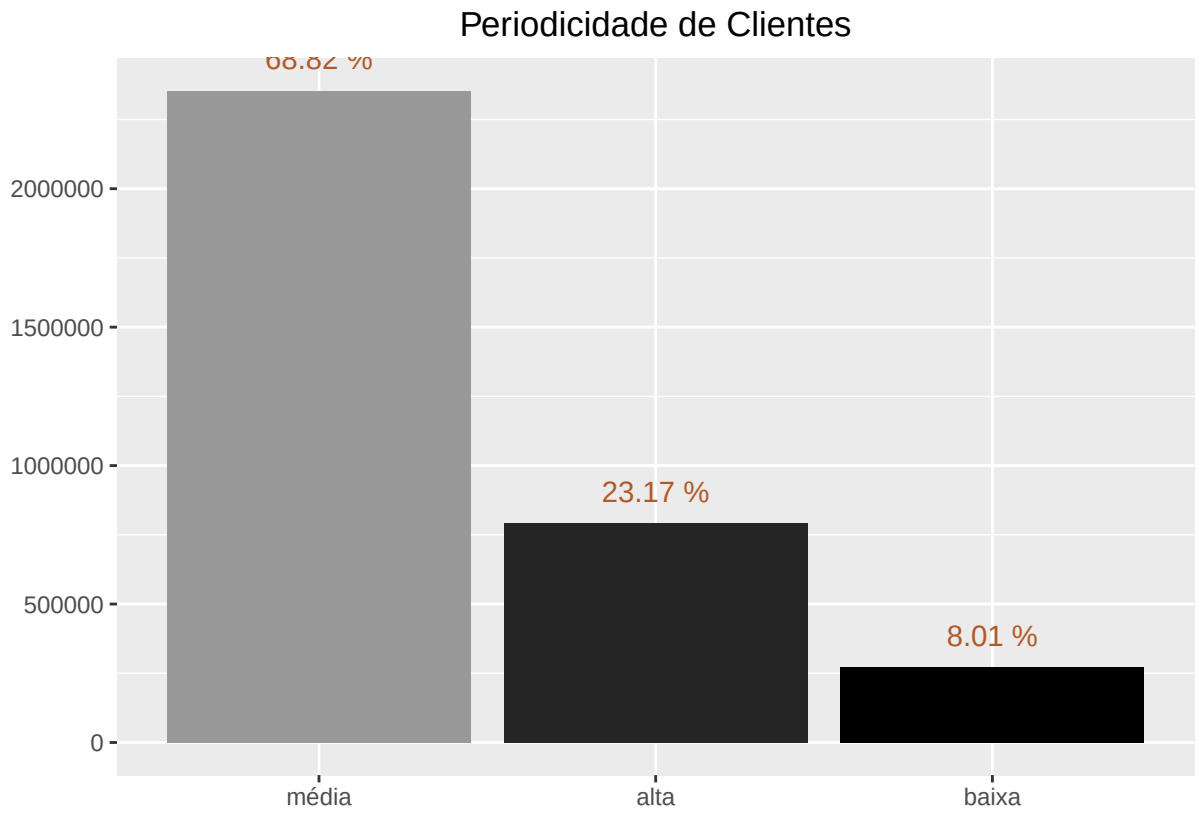
```
quadro1 <- df_pedidos %>%
    group_by(order_hour_of_day) %>%
    summarise(contagem = n())

media1 <- round(mean(c(quadro1$contagem[quadro1$order_hour_of_day <= 15],
                quadro1$contagem[quadro1$order_hour_of_day <= 15])),0)

df_pedidos %>%
    group_by(order_hour_of_day) %>%
    summarise(contagem = n()) %>%
    ggplot(aes(x = order_hour_of_day, y = contagem, group = 1)) + geom_line(size = 1) +
    geom_text(aes(label = paste("média no pico =", media1), x = 12.7, y = 255000),
              colour = "#B15928", size = 4) +
    labs(title ='Quantidade de Pedidos por Hora', x = '', y = '') +
    scale_x_continuous(labels = paste0(seq(0, 25, 5), 'h')) +
    scale_y_continuous(limits = c(5000, 300000), breaks = seq(5000, 300000, 50000)) +
    theme(plot.title = element_text(hjust = 0.5))
```



Percebemos um pico de pedidos entre 10h e 15h, que se mantém estável e inicia queda por volta das 16h. Vale atentar para reposição de produtos, atendimento e instabilidades do sistema neste período de tempo.

Gráfico de Periodicidade

```r
df_pedidos %>%
    group_by(periodicidade) %>%
    summarise(contagem = n()) %>%
    mutate(percentual = paste(round(contagem / sum(contagem) * 100,2), "%")) %>%
    ggplot(aes(x = reorder(periodicidade, - contagem), y = contagem)) +
    geom_col(aes(fill = periodicidade)) +
    geom_text(aes(label = percentual), vjust = -1, colour = "#B15928") +
    labs(title ='Periodicidade de Clientes', x = '', y = '') +
    theme(plot.title = element_text(hjust = 0.5), legend.position="none") +
    scale_fill_manual(values = cores[8:10])
```

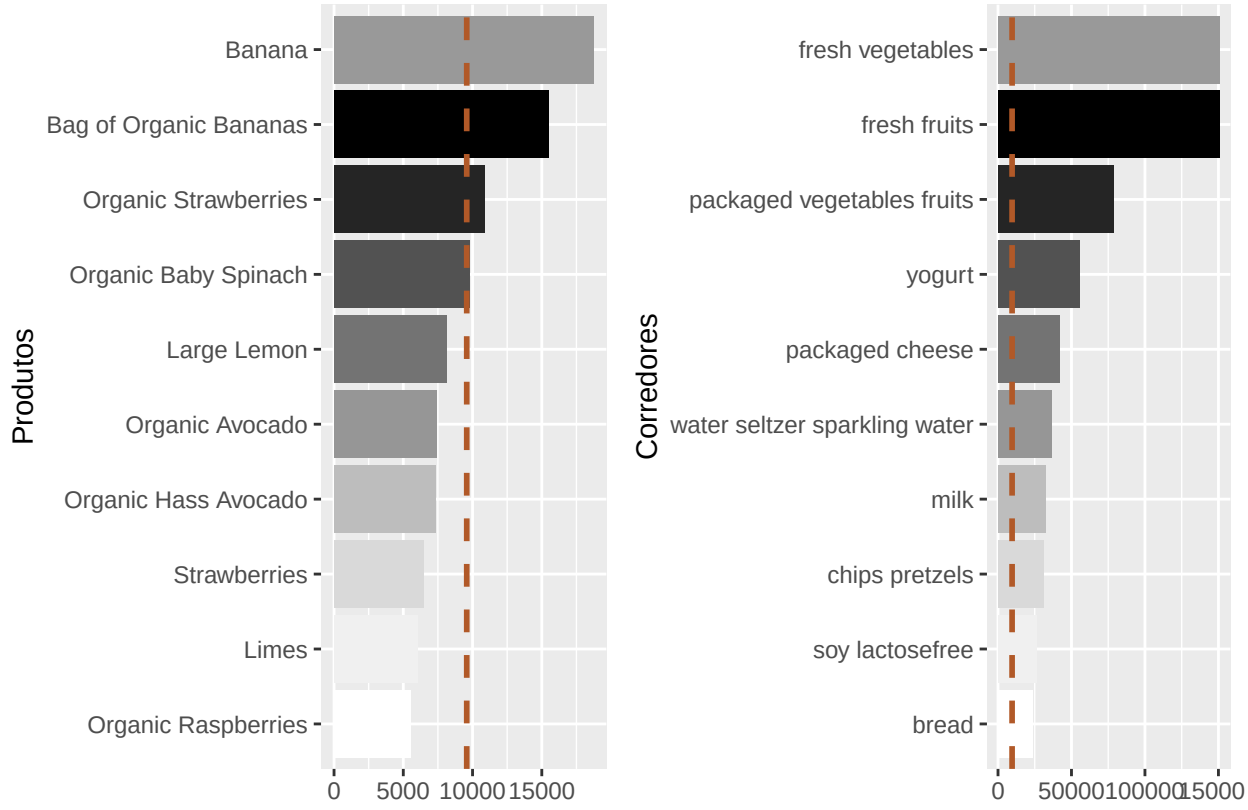


Em sua maioria, de acordo com o conceito periodicidade que definimos anteriormente, o perfil dos usuários deste conjunto de dados é de consumo médio, ou seja, realizaram de 1 a 15 pedidos no período de tempo analisado. Outro aspecto a ressaltar é que os clientes com periodicidades "extremas" (alta e baixa) correspondem a menos de 32% do total de clientes deste dataset.

Gráfico de Corredores e Produtos mais frequentes

```r
quadro_graf1 <- pedidos_esp %>%
    left_join(produtos, by = "product_id") %>%
    group_by(product_name) %>%
    summarise(contagem = n()) %>%
    mutate(product_name = reorder(product_name, contagem)) %>%
    slice_max(product_name, n = 10)

media_graf1 <- mean(quadro_graf1$contagem)
```

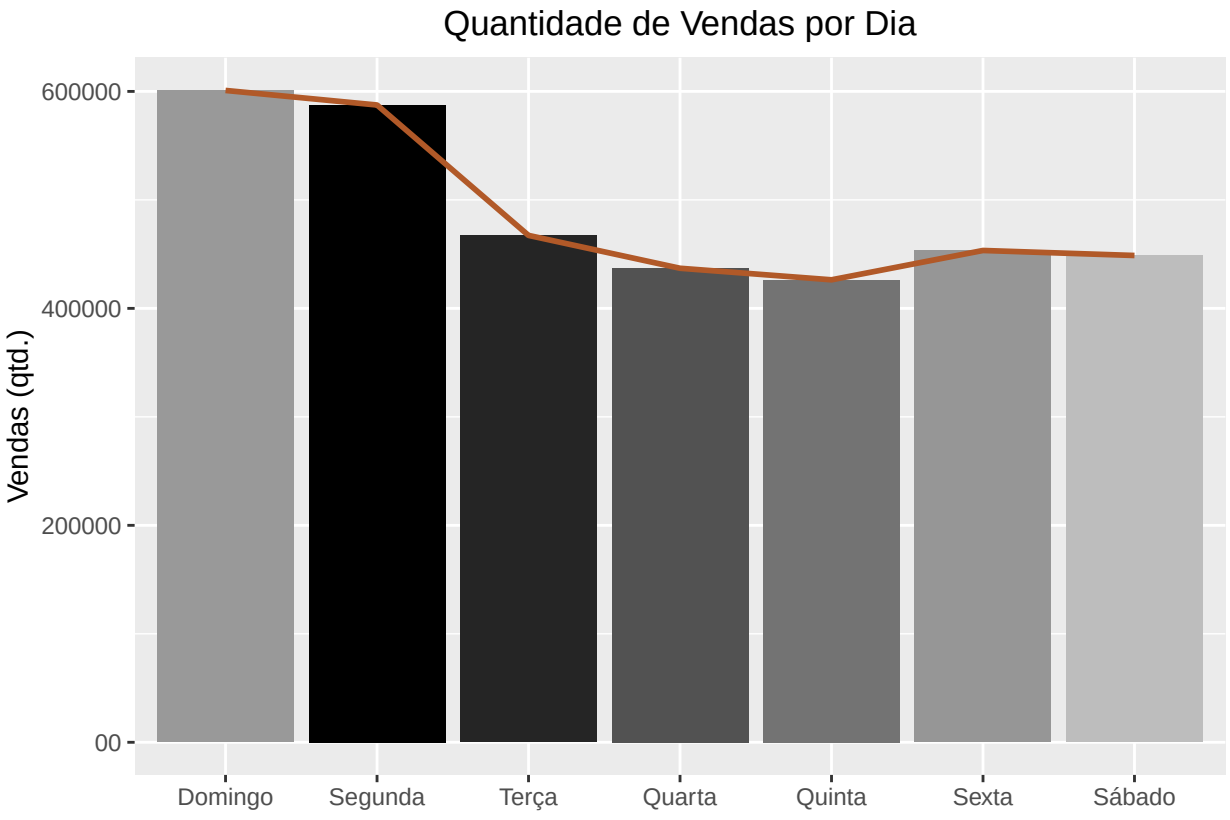
```r
graf1 <- pedidos_esp %>%
    left_join(produtos, by = "product_id") %>%
    group_by(product_name) %>%
    summarise(contagem = n()) %>%
    mutate(product_name = reorder(product_name, contagem)) %>%
    slice_max(product_name, n = 10) %>%
    ggplot(aes(x = product_name, y = contagem)) + geom_col(aes(fill = product_name)) +
    geom_hline(yintercept = media_graf1, linetype = "dashed", size = 1, colour = "#B15928") +
    coord_flip() + labs(x = 'Produtos', y = '') +
    theme(legend.position="none") +
    scale_fill_manual(values = cores)

quadro_graf2 <- pedidos_esp %>%
    left_join(produtos, by = "product_id") %>%
    left_join(corredores, by = "aisle_id") %>%
    group_by(aisle) %>%
    summarise(contagem = n()) %>%
    mutate(aisle = reorder(aisle, contagem)) %>%
    slice_max(aisle, n = 10)

media_graf2 <- mean(quadro_graf1$contagem)

graf2 <- pedidos_esp %>%
    left_join(produtos, by = "product_id") %>%
    left_join(corredores, by = "aisle_id") %>%
    group_by(aisle) %>%
    summarise(contagem = n()) %>%
    mutate(aisle = reorder(aisle, contagem)) %>%
    slice_max(aisle, n = 10) %>%
    ggplot(aes(x = aisle, y = contagem)) + geom_col(aes(fill = aisle)) + coord_flip() +
    geom_hline(yintercept = media_graf2, linetype = "dashed", colour = "#B15928", size = 1) +
    labs(x = 'Corredores', y = '') +
    theme(legend.position="none") +
    scale_fill_manual(values = cores)

grid.arrange(graf1, graf2, ncol = 2)
```



Os produtos mais pedidos neste conjunto de consumidores são frutas. Logicamente, os corredores mais visitados são, prioritariamente, os de frutas e vegetais frescos e orgânicos, seguidos de outros corredores nos quais você pode encontrar este tipo de produto ou similares (iogurtes e queijos) Outro ponto importante é que, observando o gráfico de produtos, apenas os 4 primeiros ultrapassaram a média de quantidade de compras, já em relação ao gráfico de corredores, todos os 10 primeiros pelo menos foram acessados mais do que a média da selação.

Gráfico de Qtd de vendas por dia

```
quadro2 <- df_pedidos %>%
    group_by(dia_da_semana) %>%
    summarise(contagem = n() / 10)

#min(quadro2$contagem)
#max(quadro2$contagem)

df_pedidos %>%
    group_by(dia_da_semana) %>%
    summarise(contagem = n()/10) %>%
    ggplot(aes(x = dia_da_semana, y = contagem)) + geom_col(aes(fill = dia_da_semana)) +
    geom_line(aes(y = contagem, group = 1), size = 1, colour = "#B15928") +
    labs(title ='Quantidade de Vendas por Dia', x = '', y = 'Vendas (qtd.)') +
    scale_y_continuous(name = "Vendas (qtd.)", label = paste0(seq(0, 60100, 20000), '0'),
                       limits = c(0, 60100), breaks = seq(0, 60100, 20000)) +
    theme(plot.title = element_text(hjust = 0.5), legend.position="none") +
    scale_fill_manual(values = cores[10:4])
```



De acordo com os dados deste dataset, há mais pedidos no domingo e na segunda, enquanto nos próximos dias parece haver uma média constante, retomando levamente o crescimento na sexta.

Mapa de palavras

```
df_wc <- pedidos_esp %>%
    left_join(produtos, by = "product_id") %>%
    group_by(product_name) %>%
    summarise(soma = sum(reordered)) %>%
    mutate(product_name = reorder(product_name, soma)) %>%
    slice_max(product_name, n = 200)

wordcloud(words = df_wc$product_name, freq = df_wc$soma, scale = c(6,1),
          random.order = FALSE, random.color = FALSE , rot.per = 0.35,
          ordered.colors = TRUE, colors = rep(cores[10:6], 40))
```

Observamos o mesmo padrão, então, neste mapa de palavras, encontramos com bastante frequência produtos do tipo frutas, legumes e verduras. O que vemos de diferente são alguns produtos que diferem levemente, mas naõ totalmente do grupo observado anteriormente, como leite, macarrão e manteiga. Aparentemente as pessoas realizam compras para cozinhar refeições, nas quais necessitam produtos complementares, além dos originais.

Tabela de produtos que mais foram adicionados primeiro ao carrinho

```
prod_table <- pedidos_esp %>%
    filter(add_to_cart_order == 1) %>%
    left_join(produtos, by = "product_id") %>%
    group_by(product_name) %>%
    summarise(contagem = n()) %>%
    mutate(product_name = reorder(product_name, contagem)) %>%
    slice_max(product_name, n = 20)

prod_table %>%
    rename("Produto" = "product_name", "Frequência" = "contagem") %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

Produto	Frequência
Banana	4605
Bag of Organic Bananas	3889
Organic Whole Milk	1144
Organic Avocado	995
Organic Strawberries	900
Organic Baby Spinach	869
Organic Hass Avocado	797
Spring Water	730
Strawberries	707
Sparkling Water Grapefruit	647
Large Lemon	575
Soda	557
Organic Raspberries	557
Hass Avocados	452
Raspberries	435
Large Alfresco Eggs	405
Organic Half & Half	397
Organic Blueberries	393
Organic Banana	384
Unsweetened Almondmilk	378

Nesta tabela observamos, também, a presença de alimentos específicos e também perecíveis, ou seja, de pronto consumo. Isso pode significar que o consumidor, alem de se preocupar com o preparo de suas refeições e, ao realizar as compras, pretende elaborar seus pratos de maneira imediata.

Aplicação dos Algoritmos de MBA (ECLAT e APRIORI)

Join com dataset de produtos para coletar nome de cada produto

```
df_modelo <- pedidos_esp %>%
    left_join(produtos, by = "product_id")
```

Algoritmos de Regra de Associação e seus detalhes O que é um algoritmo de regra de associação? * Suporte = conta quantas vezes o item aparece na relação de proutos * Confiança = é o cálculo da probabilidade da ocorrência de uma dupla de produtos * Elevação = é a medida que informa quanto um item X aumenta a possibilidade de ter o item Y no mesmo pedido. Para elevações > 1, sabemos que ter X no pedido leva também a pedir Y, já elevações < 1 nos informa que pedir X não significa aumento de chances de pedir também Y, mesmo que a confiança entre os 2 itens esteja alta.

Transformação Estes algoritmos esperam receber uma lista de listas para cada transação (pedido), então, realizarei um agrupamento de acordo com o id do pedido, apresentando todos os itens comprados em cada pedido:

```
df_modelo <- df_modelo %>%
    group_by(order_id) %>%
    summarize(produtos = as.vector(list(product_name)))
```

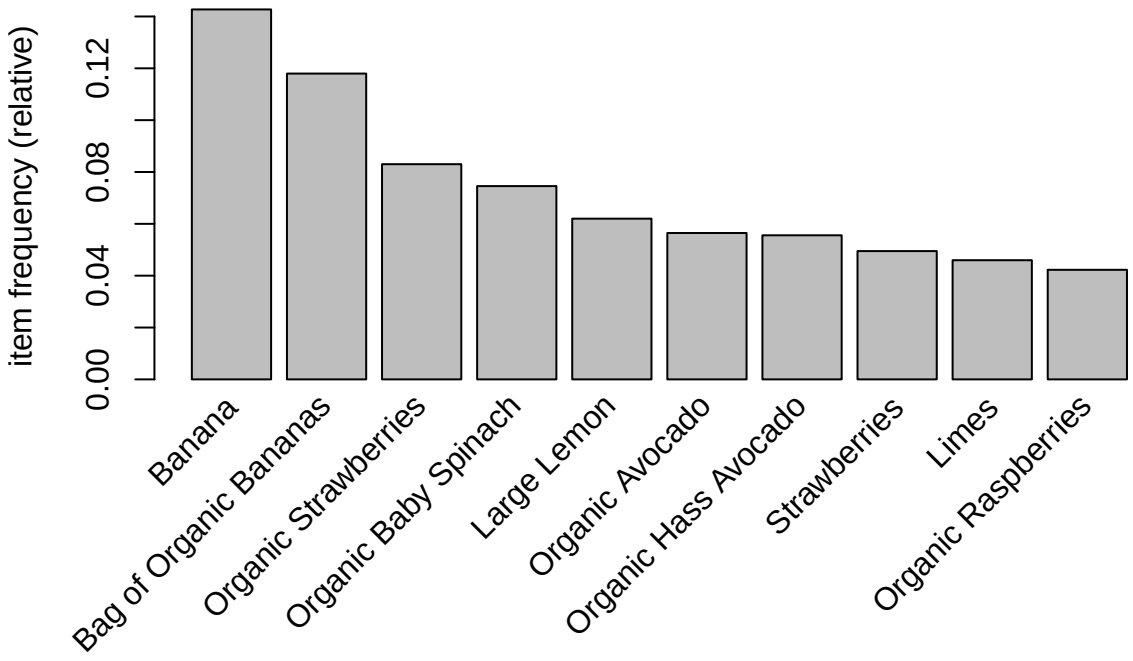
Transformando o dataset em um objeto do tipo transactions, formando uma matriz esparsa e entendendo o resultado:

```
transacoes <- as(df_modelo$produtos, "transactions")
summary(transacoes)
```

```
## transactions as itemMatrix in sparse format with
##  131209 rows (elements/itemsets/transactions) and
##  39123 columns (items) and a density of 0.0002697329
##
## most frequent items:
##                 Banana Bag of Organic Bananas   Organic Strawberries
##                  18726                  15480                  10894
##   Organic Baby Spinach            Large Lemon                (Other)
##                   9784                   8135                1321598
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 6845 7368 8033 8218 8895 8708 8541 7983 7217 6553 6034 5383 4843 4394 3831 3522
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
## 3108 2719 2473 2102 1857 1681 1462 1292 1079  986  860  679  634  553  446  403
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
##  346  315  280  210  193  178  142   99   90   88   75   79   64   48   49   32
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
##   26   31   24   23   18   15   12   10    6    5    4    8    3    3    5    4
##   65   66   67   68   70   72   74   75   76   77   80
##    3    2    1    2    4    2    2    1    2    1    2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    5.00    9.00   10.55   14.00   80.00
##
## includes extended item information - examples:
##                         labels
## 1            #2 Coffee Filters
## 2 #2 Cone White Coffee Filters
## 3        #2 Mechanical Pencils
```

Plot da distribuição de frequência dos 10 itens que mais aparecem

```
itemFrequencyPlot(transacoes, topN = 10)
```



O resultado dessa transformação foi um elemento do tipo "transactions" com 131209 linhas (itemset) e 39123 colunas (items), sendo os itens mais frequentes: Banana, Bolsa de Bananas Orgânicas e Morangos Orgânicos.

ECLAT O algoritmo ECLAT lida apenas com o suporte, ou seja, leva em consideração apenas a frequência dos itens de acordo com as transações (pedidos, neste caso), sendo uma versão simplificada do algoritmo APRIORI.

```
df_eclat %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

	items	support	transIdenticalToItemsets	count
[1]	{Bag of Organic Bananas,Organic Strawberries}	0.0234283	3074	3074
[2]	{Bag of Organic Bananas,Organic Hass Avocado}	0.0184439	2420	2420
[3]	{Bag of Organic Bananas,Organic Baby Spinach}	0.0170415	2236	2236
[4]	{Banana,Organic Avocado}	0.0168891	2216	2216
[5]	{Banana,Organic Strawberries}	0.0165690	2174	2174
[6]	{Banana,Large Lemon}	0.0164470	2158	2158
[7]	{Banana,Organic Baby Spinach}	0.0152429	2000	2000
[8]	{Banana,Strawberries}	0.0148465	1948	1948
[9]	{Bag of Organic Bananas,Organic Raspberries}	0.0135661	1780	1780
[10]	{Organic Raspberries,Organic Strawberries}	0.0127278	1670	1670

Observamos a presença do item banana (orgânica ou não) em quase todas as 10 primeiras regras. Isso significa que esse item é um dos mais badalados e o mercado deve sempre estar atento ao estoque e reposição

nas gôndulas. Vale também observar os horários de pico de compras para que nunca falte este item para os consumidores.

APRIORI A diferença do APRIORI para o ECLAT é a inclusão do parâmetro de confiança e da elevação no cálculo, tornando este um algoritmo mais completo.

```
df_apriori <- data.frame(
    lhs = labels(lhs(combinacoes_apriori)),
    rhs = labels(rhs(combinacoes_apriori)),
    combinacoes_apriori@quality)[1:10,]

df_apriori %>%
  mutate(sup = round(support, 3)) %>%
  mutate(con = round(confidence,2)) %>%
  mutate(cov = round(coverage,2)) %>%
  mutate(lift = round(lift,2)) %>%
  select(-support, -confidence, -coverage) %>%
  select(lhs, rhs, sup, con, lift, cov, count) %>%
  kbl() %>% kable_paper('striped',full_width = F) %>%
  row_spec(0, bold = T) %>%
  column_spec(1:2, width = "5cm")
```

lhs	rhs	sup	con	lift	cov	count
{Bartlett Pears}	{Banana}	0.004	0.39	2.71	0.01	466
{Granny Smith Apples}	{Banana}	0.003	0.31	2.21	0.01	434
{Yellow Bell Pepper}	{Orange Bell Pepper}	0.003	0.34	23.91	0.01	396
{Unsweetened Original Almond Breeze Almond Milk}	{Banana}	0.003	0.30	2.09	0.01	437
{Clementines, Bag}	{Banana}	0.004	0.32	2.21	0.01	466
{Red Raspberries}	{Banana}	0.003	0.29	2.06	0.01	439
{Grape White/Green Seedless}	{Banana}	0.003	0.26	1.84	0.01	402
{Organic Broccoli}	{Bag of Organic Bananas}	0.004	0.32	2.69	0.01	525
{Organic Grade A Free Range Large Brown Eggs}	{Bag of Organic Bananas}	0.003	0.25	2.13	0.01	445
{Organic Black Beans}	{Bag of Organic Bananas}	0.003	0.26	2.22	0.01	412

lhs = left-hand side rhs = right-hand side Ou seja, no quadro vemos os itens que foram associados. Se analisarmos a primeira dupla, observamos que quando o consumidor compra Bartlett Pears, existe 38% de chances dele também levar Banana (confidence). Se analisarmos, porém, a terceira dupla deste dataset, observamos que quando se compra Yellow Bell Pepper, são 24 vezes mais propensos a comprar Orange Bell Pepper (lift = 23,91%).

Conclusões Parte 1 De forma geral, frutas e verduras são os itens mais comprados e, geralmente, são comprados juntos. Cabe ao mercado organizar esses produtos nas gôndulas e corredores de uma maneira que facilite ao consumidor realizar suas compras. Aparentemente os compradores parecem ter gosto por cozinhar, comprando itens para o preparo das refeições. Também observamos bebidas leves e saudáveis como águas saborizadas. Aparentemente os usuários possuem hábitos de vida saudáveis.

Parte 2

Análise Exploratória com fins de previsão de valores Fazendo previsões de acordo com o user_id ou order_id - quais os produtos que serão comprados juntos?

Join dos datasets e retiradas as colunas de dia da semana e hora da realização do pedido e número do pedido pois não agregam valor a esta parte do projeto:

```
novo_df <- df_pedidos2 %>%
    select(-order_dow, -order_hour_of_day, -order_number) %>%
    left_join(pedidos_esp, by = c("order_id"))
```

Feature Engineering

Quantidade de produtos naquela compra

```
novo_df <- novo_df %>%
    left_join(novo_df %>% group_by(order_id) %>% summarize(qtd_prod_por_pedido = n()),
              by = c ("order_id"))
```

Quantidade de vezes que o produto foi pedido

```
novo_df <- novo_df %>%
    left_join(novo_df %>% group_by(product_id) %>% summarize(qtd_vezes_prod_pedido = n()),
              by = c("product_id"))
```

Quantidade de vezes que o produto foi pedido como 1 item do carrinho

```
novo_df <- novo_df %>%
    left_join(novo_df %>% filter(add_to_cart_order == 1) %>%
                          group_by(product_id) %>% summarize(prim_carrinho = n()),
              by = c("product_id"))
```

Quantidade de vezes que o produto foi reorded

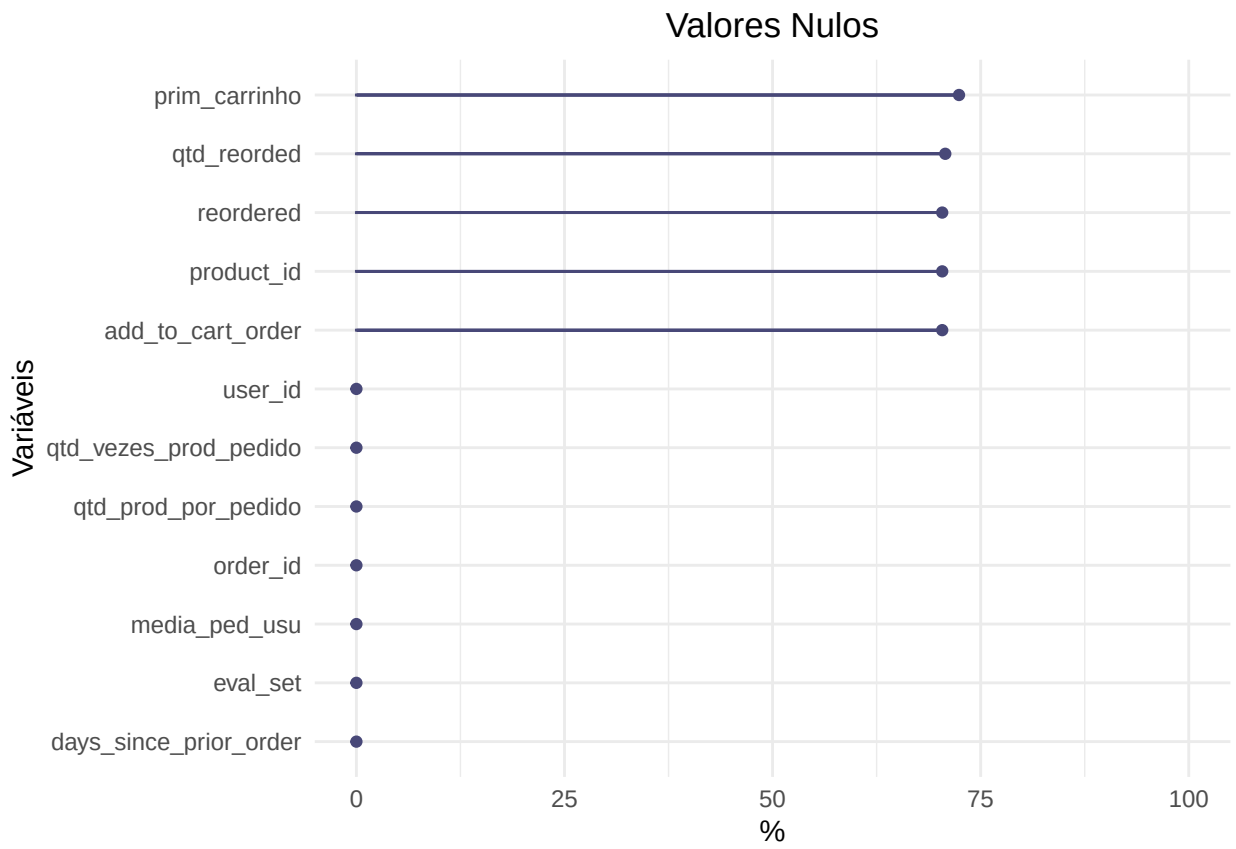
```
novo_df <- novo_df %>%
    left_join(novo_df %>% filter(reordered == 1) %>%
                          group_by(product_id) %>% summarize(qtd_reorded = n()),
              by = c("product_id"))
```

Quantidade de pedidos feitos pelo usuário / Média total de pedidos

```
novo_df <- novo_df %>%
    left_join(novo_df %>% group_by(user_id) %>% summarize(contagem = n())
              %>% mutate(media_ped_usu = round(contagem / mean(contagem),2)) %>%
                  select(-contagem),
              by = c("user_id"))
```

Substituindo valores NA

```
gg_miss_var(novo_df, show_pct = TRUE) +
    labs(x = "Variáveis", y = "%", title = "Valores Nulos") +
    theme(plot.title = element_text(hjust = 0.5)) + ylim(0, 100)
```



As colunas prim_carrinho, qtd_reorded e add_to_cart_order serão preenchidas com zeros e as colunas reorded e product_id se manterão com NAs pois serão deletadas do dataset antes do início do treinamento do modelo.

```
novo_df$prim_carrinho[is.na(novo_df$prim_carrinho)] <- 0
novo_df$qtd_reorded[is.na(novo_df$qtd_reorded)] <- 0
novo_df$add_to_cart_order[is.na(novo_df$add_to_cart_order)] <- 0
```

Conferindo se temos apenas reorded e product_id com valores nulos:

```
df_nulo <- as.data.frame(sapply(novo_df, function(y) sum(length(which(is.na(y))))))
names(df_nulo) <- "Soma de Nulos"
df_nulo %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

	Soma de Nulos
order_id	0
user_id	0
eval_set	0
days_since_prior_order	0
product_id	3289874
add_to_cart_order	0
reordered	3289874
qtd_prod_por_pedido	0
qtd_vezes_prod_pedido	0
prim_carrinho	0
qtd_reorded	0
media_ped_usu	0

Treinamento e Avaliação do Modelo

Divisão do dataset onde eval_set == 'train' entre treino e teste

```
treino <- novo_df %>%
    filter(eval_set == "train") %>%
    select(-order_id, -eval_set)

divisao <- createDataPartition(treino$reordered, p = .8, list = FALSE)
df_treino <- treino[divisao,]
df_teste <- treino[-divisao,]
```

Criação de matriz xgb e criação do Modelo

Aplicação do modelo no grupo de teste

```
mat_teste <- xgb.DMatrix(as.matrix(df_teste %>% select(-reordered, -product_id, -user_id))
                         ,label = df_teste$reordered)

df_teste$previsoes <- predict(modelo, mat_teste)
```

Como o algoritimo xgboost nos entrega uma probabilidade, ou seja, se a previsão foi de 62%, isto significa que existe 62% de chances daquele produto ser pedido novamente, vamos aplicar um limiar (threshold) para selecionar apenas previsões >= 70%

```
df_teste$previsoes <- ifelse(df_teste$previsoes >= .7, 1, 0)
```

Avaliação do modelo Para saber se o modelo foi bem sucedido, vou verificar se as previsoes acertaram se comparadas às verdadeiras classificações do dataset

```
df_teste$sucesso <- ifelse(df_teste$reordered == df_teste$previsoes, "sim", "não")

df_teste %>%
    group_by(sucesso) %>%
    summarize(contagem = n()) %>%
    mutate(`% de acertos` = paste0(round(((contagem / sum(contagem)) * 100), 2), " %")) %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T)
```

sucesso	contagem	% de acertos
não	95574	34.51 %
sim	181349	65.49 %

Nosso modelo acertou pouco mais 65% dos casos, não sendo muito satisfatório. Vamos tentar alcançar pelo menos 70% com um novo treinamento com mais rounds:

```
df_teste$previsoes2 <- predict(novo_modelo, mat_teste)

df_teste$previsoes2 <- ifelse(df_teste$previsoes2 >= .7, 1, 0)
```

Agora, para avaliar o modelo, criaremos uma matriz de confusão A matriz de confusão é uma tabela de contingência de tamanho 2 x 2 que nos permite identificar quantas vezes o modelo acertou e quantas ele errou

```
confusionMatrix(as.factor(df_teste$previsoes2), as.factor(df_teste$reordered))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 93819 77866
##          1 17142 88096
##
##                Accuracy : 0.6569
##                  95% CI : (0.6551, 0.6587)
##     No Information Rate : 0.5993
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.345
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8455
##             Specificity : 0.5308
##          Pos Pred Value : 0.5465
##          Neg Pred Value : 0.8371
##              Prevalence : 0.4007
##          Detection Rate : 0.3388
##    Detection Prevalence : 0.6200
##       Balanced Accuracy : 0.6882
##
##        'Positive' Class : 0
##
```

Não obtivemos resultado muito diferente, a acurácia segue mais ou menos por volta de 65%. Na matriz de confusão observamos poucos falsos positivos - quantidade de vezes que o produto não foi pedido novamente mas o modelo previu que sim - mas altíssimos casos de falsos negativos - quantidade de vezes que o produto foi pedido novamente mas o modelo previu que não.

Conclusões Parte 2

Não obtive bons resultados com a aplicação do algoritmo xgboost. Cabe realização de mais estudos para aplicações de outros algoritmos ou, a realização de nova etapa de feature engineering ou, ainda, o ajuste de hiperparâmetros no algoritmo usado.

Entrega do Resultado - quais produtos previstos para os usuários em seu próximo pedido?

Antes de entregar o resultado, vamos comparar os produtos que de fato foram comprados e os que nosso modelo previu

```
previsoes <- df_teste %>%
    filter(previsoes2 == 1) %>%
    inner_join(produtos, by ="product_id") %>%
    group_by(user_id) %>%
    summarize(produtos_previstos = paste0(product_name, collapse = ", "))

produtos_reais <- df_teste %>%
    filter(reordered == 1) %>%
    inner_join(produtos, by ="product_id") %>%
    group_by(user_id) %>%
    summarize(produtos_reais = paste0(product_name, collapse = ", "))

comparando_produtos <- previsoes %>%
    inner_join(produtos_reais, by = "user_id")

comparando_produtos[1:10,] %>%
    rename("ID" = "user_id"
           ,"Previstos" = "produtos_previstos"
           ,"Reais" = "produtos_reais") %>%
    kbl() %>% kable_paper('striped',full_width = F) %>%
    row_spec(0, bold = T) %>%
    column_spec(2:3, width = "7cm")
```

ID	Previstos	Reais
2	Plantain Chips, Banana, Honeycrisp Apple, Uncured Slow Cooked Ham	Plantain Chips, Banana, Honeycrisp Apple, Pad Thai
5	Organic Large Extra Fancy Fuji Apple	Organic Grape Tomatoes, Organic Baby Arugula
7	85% Lean Ground Beef	85% Lean Ground Beef, Organic Dark Brown Sugar
13	Whole Milk	Whole Milk
14	Seasoned Chicken Fry Batter Mix	Seasoned Chicken Fry Batter Mix
17	Strawberries	Strawberries
21	Sparkling Water Grapefruit	Sparkling Water Grapefruit
27	Organic Strawberries, Seedless Red Grapes, Total 2% with Strawberry Lowfat Greek Strained Yogurt, Organic Raspberry Lowfat Yogurt	Organic Strawberries, Seedless Red Grapes, Total 2% with Strawberry Lowfat Greek Strained Yogurt
29	Original 0 Calorie Sweetener Packets 115 Count	Original 0 Calorie Sweetener Packets 115 Count, Mini Oreo Go Pak, Pure Life Purified Water
42	Organic Blueberries, Organic White Chocolate Peanut Butter Cups	Original Chicken Patties, Organic Blueberries, Organic White Chocolate Peanut Butter Cups

Salvando o Resultando

```r
sub <- df_teste %>%
    filter(previsoes2 == 1) %>%
    group_by(user_id) %>%
    summarize(produtos = paste0(product_id, collapse = " "))

write.csv(sub, file = "datasets/resultado_final.csv", row.names = F, quote = F)
```

Referências Bibliográficas

- "The Instacart Online Grocery Shopping Dataset 2017", accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on 2022-01-09
- Formação Cientista de Dados - Data Science Academy